

REMARKS

Claims 1-24 are presented for further examination. Claims 1-3, 5, 8-11, 15-16, and 20 have been amended.

In the Office Action mailed March 11, 2005, the Examiner rejected claims 1-11 and 15-21 under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,832,219 ("Pettus") in view of U.S. Patent No. 6,718,399 ("Chernick et al."). Claims 14 and 24 were rejected under 35 U.S.C. § 103(a) as unpatentable over Pettus and Chernick et al. in view of "Official Notice." Claims 12 and 22 were rejected under 35 U.S.C. § 103(a) as unpatentable over Pettus and Chernick et al. in view of U.S. Patent No. 6,182,068 ("Lomet et al."). Claims 13 and 23 were rejected under 35 U.S.C. § 103(a) as unpatentable over Pettus and Chernick et al. in view of U.S. Patent No. 6,223,205 ("Harchol-Balter et al.").

Applicants have amended claims 1, 5, 10, 15 and 20 to recite that a client component requests an interface manager to invoke a server component interface method on behalf of the client component. Rather than invoking the interface methods directly by sending requests to individual server components, the client component sends all requests to invoke interface methods of server components to a single entity, the interface manager. This method of interaction between the client component and the interface manager, and between the interface manager and the server components, is clearly different than that described by either Pettus or Chernick et al..

More particularly, Pettus describes an object-oriented Remote Procedure Call (RPC) implementation. A client object of Pettus invokes server object methods directly rather than sending requests to invoke server object methods to an interface manager. The client object sends requests to invoke server object methods of multiple service objects to each individual service object.

Pettus requires a caller object residing in the client node to intercept the client object request and pass the request across a network to a dispatcher object residing in a server node. The dispatcher object calls the method of a service program. The operation of the caller object and dispatcher object are transparent to the client object and the server object. Pettus

“provide[s] a transparent mechanism by which service requests generated by a local client object can be satisfied either by local or remote service objects” (see col. 5, lines 9-12).

Consequently, Pettus teaches away from a client component requesting that an interface manager invoke an interface method of a server component on behalf of the client component because an object of Pettus is to conceal the operation of the caller object and dispatcher object from the client object and server object. The client object of Pettus is unaware of an interface manager and therefore does not make requests of the interface manager. Instead, the client object makes requests of server objects directly.

Chernick et al. do not teach or suggest a client component requesting that an interface manager invoke an interface method of a server component on behalf of the client component. Instead, Chernick et al. teach transparency by “shield[ing] the client and server objects from the details of communications therebetween for both local and remote message calls” (see Abstract). The client object of Chernick et al. is unaware of an interface manager and therefore does not make requests of the interface manager. Instead, the client object makes requests of server objects directly.

Claims 1 and 15 have been amended to include a client component requesting that an interface manager invoke the interface method on behalf of the client component. Claims 10 and 20 have been amended to include a station manager requesting that an interface manager invoke the interface method on behalf of the client component. Claim 5 has been amended to include a first component requesting that an interface manager invoke the interface method on behalf of the client component.

As discussed above, nowhere do Pettus or Chernick et al., taken alone or in any combination thereof, teach or suggest a client component requesting that an interface manager invoke an interface method of a server component on behalf of the client component. For these reasons alone, applicants respectfully submit that independent claims 1, 5, 10, 15, and 20, as well as all claims depending therefrom, are clearly allowable.

Applicants have also amended claims 1, 5, 10, 15 and 20 as set forth above to recite that a client component, interface manager, and server components execute within the

same network device. This location of these three entities within a single device is clearly different than that described by Pettus.

More particularly, Pettus describes a caller object located in a client node and a dispatcher object located in a server node (see Fig. 6). The client node and server node are distinct devices separated by a communication channel. Pettus teaches that the dispatcher object is located in a server node, not in a client node. Some service requests may be routed locally to a service program located in the client node. However, in this case a dispatcher object is not used to route the service request. A dispatcher object is not available on the client node, since Pettus teaches that dispatcher objects are located only on server nodes. Since a dispatcher object is not used in this situation to route a service request, Pettus, in this situation, fails to use "references to the interface methods" as taught by the claimed invention since the use of function pointers and a table of function pointers in Pettus is limited to the dispatcher object in the server node. Pettus does not teach the use of function pointers in the caller object or client node.

For at least these reasons, applicants respectfully submit that independent claims 1, 5, 10, 15, and 20, as well as all claims depending therefrom, are clearly allowable.

Applicants have also amended claims 1, 5, 9, 11, and 16 as set forth above to recite a function parameter passed by an interface manager to a server component. The function parameter provides a flexible way of invoking an interface method since methods requiring input parameters may be invoked. Without the use of a function parameter, the client component would be limited to invoking interface methods that do not accept parameters. The use of function parameters in conjunction with an interface manager is not taught by Pettus. More particularly, Pettus teaches the use of parameters in conventional RPC objects, but does not teach the use of parameters with a caller object or a dispatcher object.

For at least these reasons, as well as for the reasons discussed above, applicants respectfully submit that claims 1, 5, 9, 11, and 16, as well as all claims depending therefrom, are clearly allowable.

In addition, applicants have amended claim 1 as set forth above to recite a response message, status, and second message queue. The interface manager uses the second message queue to relay response messages from server components to the client component.

The server components generate response messages in response to the completion of an interface method. Chernick et al. do not teach conveying response messages including a status via a second message queue. More particularly, Chernick et al. teach the use of a single queue to store message calls.


For at least these reasons, as well as for the reasons discussed above, applicants respectfully submit that claim 1 as well as all claims depending therefrom, are clearly allowable.

In view of the foregoing, applicants submit that all of the claims in this application are clearly allowable. In the event the Examiner finds informalities that can be resolved by telephone conference, the Examiner is urged to contact applicants' undersigned representative by telephone at (206) 622-4900 in order to expeditiously resolve prosecution of this application. Consequently, early and favorable action allowing these claims and passing this case to issuance is respectfully solicited.

The Director is authorized to charge any additional fees due by way of this Amendment, or credit any overpayment, to our Deposit Account No. 19-1090.

All of the claims remaining in the application are now clearly allowable. Favorable consideration and a Notice of Allowance are earnestly solicited.

Respectfully submitted,
SEED Intellectual Property Law Group PLLC


E. Russell Tarleton
Registration No. 31,800

ERT:jl

Enclosure:
Postcard

701 Fifth Avenue, Suite 6300
Seattle, Washington 98104-7092
Phone: (206) 622-4900
Fax: (206) 682-6031

599911_1.DOC